SC4 Data Architecture – Integration and Mapping Methodology – Version 0.1

Julian Fowler, PDT Solutions

Draft for review at the WG10 workshop on 1999-04-21/23 in London

1 Introduction

This document provides – in draft form – a description of the integration and mapping methodology that is a component of the SC4 Data Architecture. The other components of the Data Architecture documentation are:

— a description of the architecture itself (latest version: WG10 N254);

— the integration model (latest version: WG10 N220).

Other related Data Architecture PWI deliverables are:

— requirements for changes and extensions to EXPRESS (latest version WG10 N249);

— proposals for a new language "EXIST";

— demonstration of the architecture.

This document is based on the PowerPoint slide presentation distributed as WG10 N253, which will be presented in updated form as WG10 N256 at the London workshop.

This document is intended to form the basis for a standard description of the integration and mapping methods. This standard description (which is currently targeted for publication as an ISO Technical Specification) will include:

— Procedures to be followed within the integration and mapping methods ("what to do").

— Practices for analysis of requirements as stated (explicitly or implicitly) in data models or applications to be integrated and/or mapped ("how to do it").

— Practices for the use of the integration model, reference data, specification language(s) and mapping language(s) ("what tools are used, and how are they applied").

— Guidelines that describe preferred or recommended ways of applying the integration and mapping methods ("what's the best way to do this?").

The structure of the document is as follows:

— Section 1 (this section) introduces the document and describes its structure.

— Section 2 provides a summary of the requirements that the integration and mapping methodology is designed to satisfy.

— Section 3 enumerates the fundemental concepts and assumptions that underlie the methodology.

— Section 4 describes the integration method by which the integration model is extended so that it satisfies the requirements of multiple external or application data models.

— Section 5 describes the mapping method by which the structure and semantics of an external or application are related to the structure and semantics of the integration model.

— Section 6 provides an overview of the process by which a particular external or application data model is mapped to the integration model.

— Section 7 compares the SC4 Data Architecture methods with those already in use for the development of SC4 standards (mainly ISO 10303 “STEP”).

— Section 8 presents the conclusions of the document and identifies the next steps required to complete the integration and mapping methodology specification.

2 Integration and mapping methodology – requirements

The methodology described in this document is designed to meet the following requirements.

a) extending the integration model to meet new requirements;

b) selecting a “subset” of the integration model that satisfies the semantics of a particular external/application model;

c) defining the mapping(s) between the selected subset and the structure of the external/application model.

The first of these methods is applicable in the case of “voids” being identified in the integration model when an external/application data model is being mapped to it. The goal of this method is to *integrate* the necessary additional concepts, and therefore to extend the scope of information that the integration model supports.

NOTE This aspect of integration is somewhat different from the resource integration method of ISO 10303. This difference is discussed further in section 7 below.

The second and third methods are applicable when requirements exist to relate some external or application data model to the integration model. Usage cases include (but are probably not limited to):

— creating an external view on the integration model, so that shared/integrated data can be presented and used for a particular purpose;

— migrating data from legacy systems to a shared/integrated environment;

— managing data in multiple external or legacy systems in an integrated manner.

NOTE There is a strong similarity between these methods and those used for application interpretation in the development of STEP Application Protocols.

3 Fundamental concepts and assumptions

The following fundamental concepts and assumptions underlie the methods that are described in this document.

a) The SC4 data architecture has the form described in WG10 N254

b) The "core" of the data architecture is an integration model that consists of:

  1) a generic data model, and

  2) reference data (instances of the generic model).

c) The integration model has a universal context – it is neutral with respect to all external views.

d) The scope of the integration model is the "union" of the scopes of all models that have been integrated with it.

NOTE As a consequences of these fundamental concepts and assumptions, there is a key difference between the Integration Model (IM) and the Integrated Resources (IRs) of ISO 10303. The IRs contain generalizations of the semantics of the models that use them (the AIMs of Application Protocols). The IM contains *precisely* the semantics of all models that use it, by definition. Unlike the IRs, the IM is syntactically and semantically complete within its defined context and scope. Therefore, in the use of the IM "interpretation" is "subsetting" – the selection of constructs that convey precisely the semantics of the external/application data model that is being mapped to the IM.

4 Integration method

This section describes the integration method. As noted in section 2 above, this method applies when the integration model requires extension, i.e., extend the IM to include concepts not previously explicit within its context and scope. There are two distinct variants on this method, depending on the nature of the requirement and the nature of the solution as determined by the Data Architecture. These variants may be characterized as follows:

a) *context* extension – addition of new constructs to the IM that extend its context, i.e., its applicability.

b) *scope* extension – addition of new constructs to the IM that extend its scope, i.e., its contents and application.

Two essential axioms of the integration method are stated below and illustrated in Figure 1.

a) Context extensions are addressed in the structure of the IM.

b) All scope extensions are achieved through population of the IM structures.

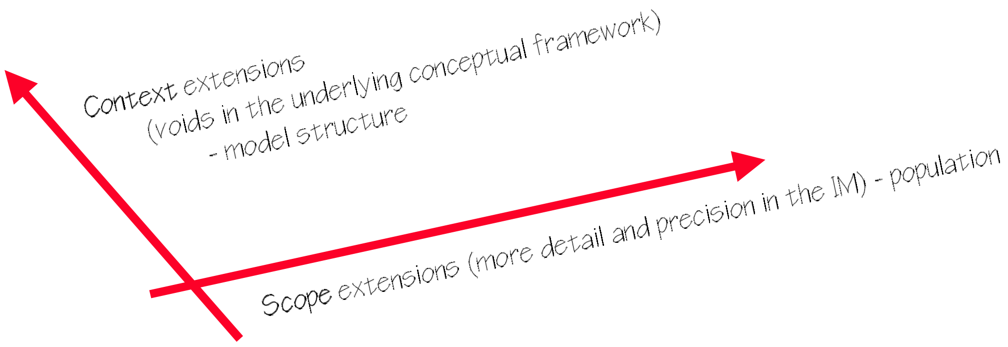
Context extensions
(voids in the underlying conceptual framework)
- model structure
Scope extensions (more detail and precision in the IM) - population

Figure 1 – context extension vs. scope extension

4.1 Context extension

The need to apply this method occurs when information is discovered that is not covered *at all* by the integration model. If the premises stated in section 3 above are correct, then this requirement may seldom be encountered (if at all). This method will be applied only when it is found that one or more *semantically irreducible* concepts are missing from the integration model, and that these concepts are to be added to the IM.

EXAMPLE Construction of a "real" example is difficult given that the IM – even it its current form – seems to include most of the concepts that will be required to integrate external/application data models in the domain of industrial data (and beyond). Consider, then, the case where the IM does not include the concepts of singular and plural individuals. These concepts cannot be expressed in terms of combination or instantiation of other components of the IM. This analysis determines that the concepts of singular and plural individuals are *semantically irreducible* and will be added to the the IM using the *context extension* method.

When such a requirement is identified, the procedure that applies is as follows.

— Based on the analysis of the requirements identify the most specific construct in the existing IM that corresponds to the semantics of the extension.

The practices that apply are as follows.

— Create a subtype[1] of the entity data type whose semantics have been identified as a generalization of the concept to be added.

— Examine any "siblings" of the new subtype and determine whether any changes to the supertype/subtype hierarchy are necessary.

The need for this second practice is illustrated in Figure 2 below.

[1] In this document I have assumed that the IM is represented in EXPRESS and that SUBTYPE is the appropriate mechanism for representing a specialization of a concept.

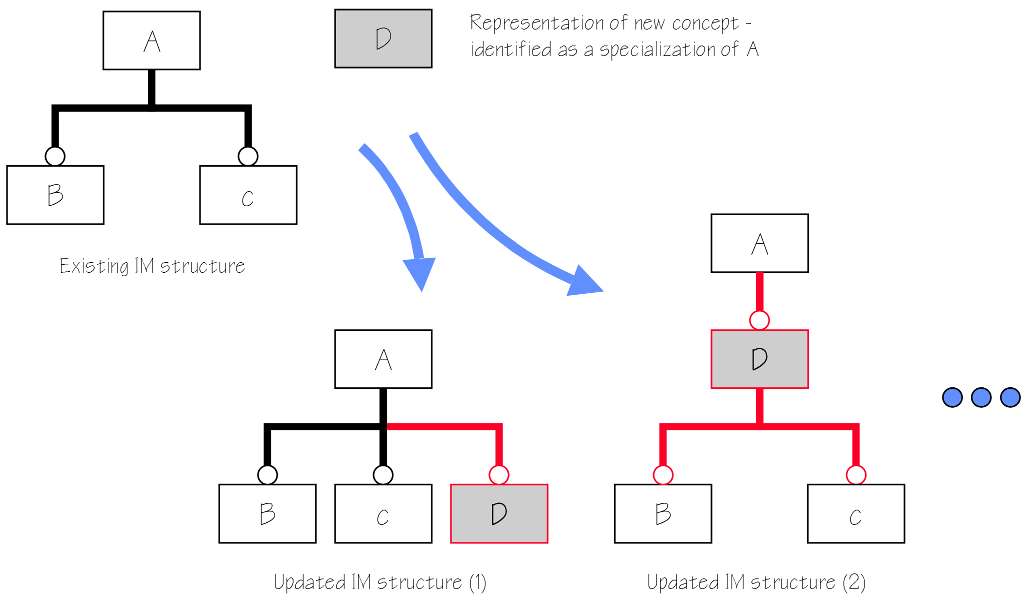

Figure 2 – sibling relationships in IM concept extension

In this diagram, **A**, **B** and **C** are entity data types in the existing IM. **D** is an entity data type that represents the concept that is to be added to the IM, and has already been identified asa specialization of **A**. The lower part of the diagram shows two possible updates to the IM structure. That labelled “Updated IM structure (1)” illustrates the case where **D** is a sibling of the existing subtypes of **A**. That labelled “Updated IM structure (2)” illustrates the case where **D** is a child of **A** but is a parent of **B** and **C**. (Clearly other combinations are possible but are not shown here.)

4.2 Scope extension

The need to apply this method occurs when the IM does not include constructs that precisely match the required semantics, but can support them by refinement (specialization and/or instantiation), or by combination of what is already there.

Issue: I’m not sure that the distinction between context extension and scope extension is as clear as it should be. This needs to be discussed at the London workshop. The issue raised below is actually related to this, as well as to the actual practice(s) to be applied.

EXAMPLE The IM contains a general concept (class) **physical_object** but not the more specific concept **aeroplane**. However, house is not a “new” concept in the integration model, but is a refinement of the existing **physical_object** concept.

When such a requirement is identified, the procedure that applies is as follows.

— Based on the analysis of the requirements identify the most specific construct in the existing IM that corre- sponds to the semantics of the extension.

(This is the same procedure that applies to the case of context extension as described above.)

The practices that apply are as follows.

Issue: some decisions are needed with respect to the practices to apply in this case. The following text discusses this issue and proposes a solution.

This kind of extension to the integration model can be achieved in one of two ways:

— through specialization of the data structure, i.e., by creation of subtypes of existing IM entity data types.

— through population of the data structure, i.e., by creation of (standard) instances of the IM entity data types.

The issue to be considered here is:

When is the model extended by specialization, and when is it extended by instantiation?

EXAMPLE In the example above, should aeroplane be mapped to a subtype of physical_object, or to an instance of class (of physical_object). This is illustrated in Figure 3 below.

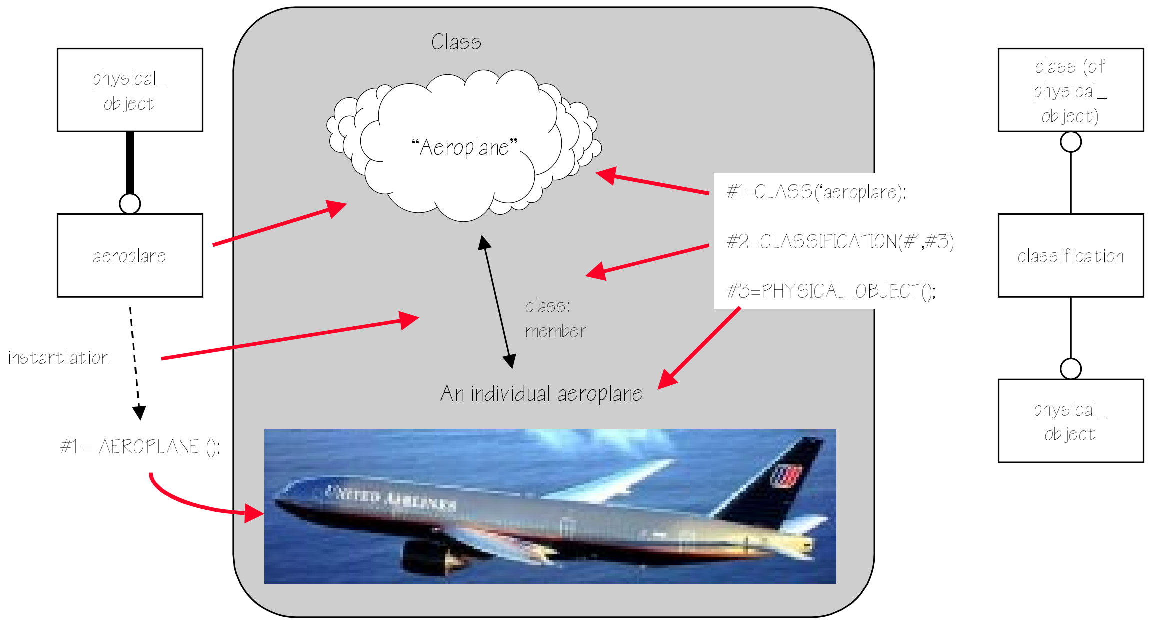

Figure 3 – specialization of the integration model by subtyping or instantiation

In this diagram, the shaded box represents the real world (and our perception and understanding of it). In this case, the things of interest in the real world are:

— an individual aeroplane (a particular Boeing 777 operated by United Airlines);

— a class whose name is "aeroplane" and whose definition is "a powered heavier-than-air flying vehicle with fixed wings";

— a link or relationship between these two things, asserting that the the first is a member of the second.

The other parts of the diagram show two possible uses of the integration model to represent this information. On the left hand side of the diagram, **aeroplane** is shown as a subtype of **physical_object**. The "real world" information is then represented as follows:

— The entity data type **aeroplane** (subtype[2] of **physical_object**) represents the class “aeroplane”.

— An instance of the entity data type **aeroplane** represents the particular aeroplane.

— The mapping between the **aeroplane** entity data type and its instance represents the class:member relationship.

On the right hand side of the diagram a different approach to representation of the same information is illustrated. Here no structural change to the integration model. Rather, the IM constructs class, classification, and individual are used directly, and the extension to the IM to represent the class “aeroplane” is achieved by creating a standard instance of class. In this case:

— An instance of the entity data type **class** represents the class “aeroplane”.

— An instance[3] of the entity data type **physical_object** (that is classified as being an aeroplane) represents the particular aeroplane.

— An instance of the **classification** entity data type represents the class:member relationship.

The second case in the example above follows axiom (b) stated in section 4 above. This axiom can be restated as follows.

> The integration model contains as data model elements (EXPRESS entity data types) a minimal set of concepts using which all other facts can be expressed as data. All scope extensions (integration) are achieved through population of the IM structures.

One of the consequences of this approach is that if we use EXPRESS for the IM structure, we have to use something like Part 21 or EXPRESS-I to represent populations of the IM. This may not be natural for data modellers in SC4, whose bias will be towards EXPRESS representation. (This is part of the discussion with respect to the use of EXPRESS in the data architecture, and relates to use of other languages such as UML, EXIST, etc.)

5 Mapping method

Figure 4 below shows the overall data integration architecture.

NOTE See N254 for a detailed explanation.

This illustrates the separation of the mapping between an external/application data model and the integration model. This separation exists at both the level of process and deliverables.

[2] There is a need to discuss this particular approach – this is the issue of whether, in this case, **aeroplane** is a *subtype* of **physical_object**, or whether it is a *member* of **physical_object**.

[3] In the Part 21 instance fragment, I have assumed that **class** has a “local” attribute for its name. In fact, within the IM other entity data types are used to associate an identifier with the instance of **class**.

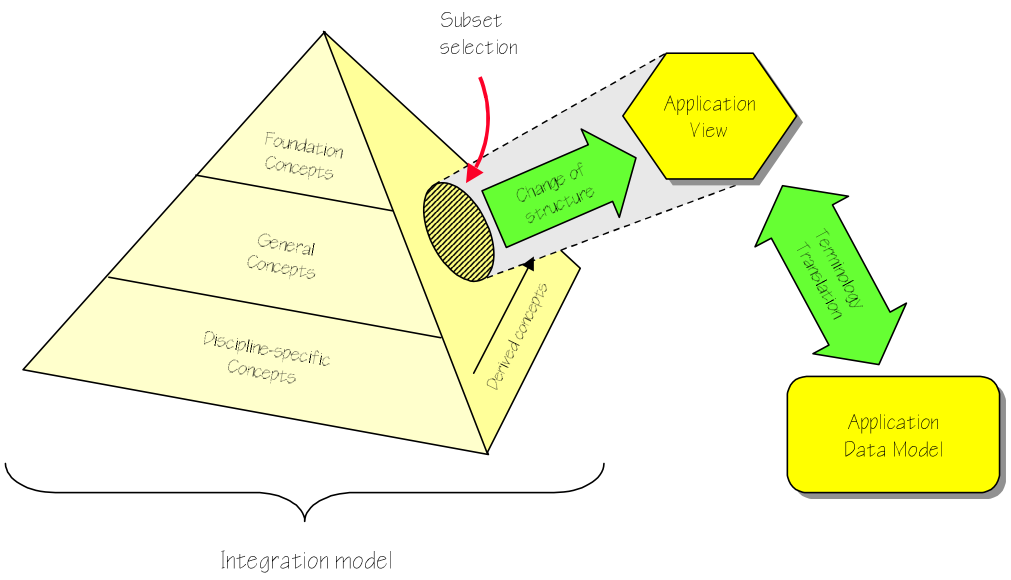

Figure 4 – integration architecture

The requirement that the mapping method addresses is the specification of the transformations between a "sub-set" of the IM and an external/application data model. These transformations include:

— structural changes;

— terminology changes;

but exclude:

— semantic changes.

NOTE This is *precisely* the relationship between the ARM and the AIM in a STEP AP – the role of the AIM is to express the semantics of the ARM in the structure and terminology of the IRs.

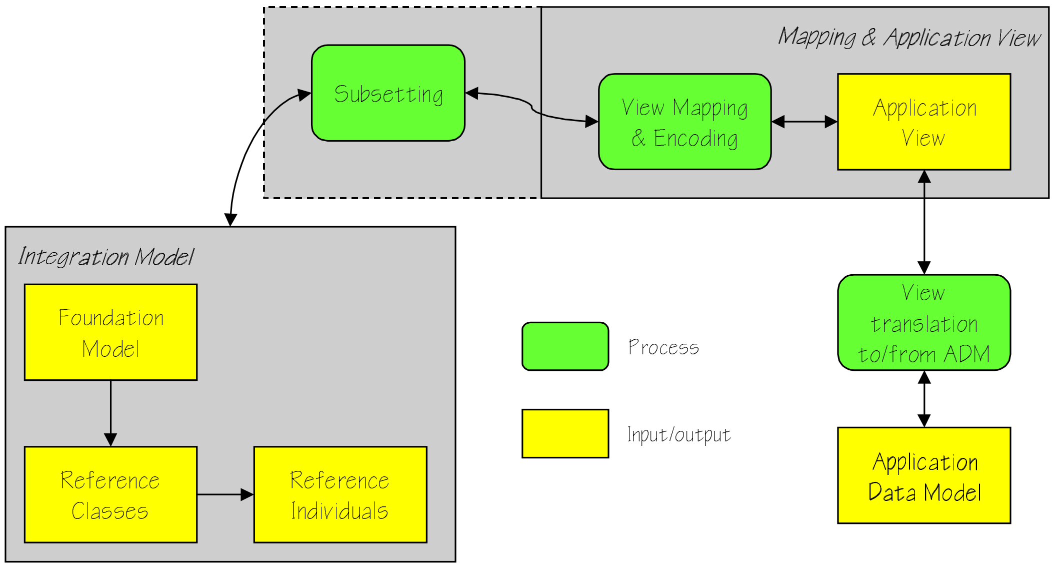

Figure 5 – relationships between the elements of the data architecture

Figure 5 (adapted from N254) makes the intended method for mapping clearer. The three aspects of method shown here are as follows.

— Identification of the subset of the IM that supports the semantics of the application (data model).

— Transformation of the structure of the subset to that of the application data model (including encoding mechanisms such as the concatenation of discrete data elements in the IM into “simple” attributes in the ADM). The result of this process is characterized here as an “application view”.

— Translation of the application view to the application data model (terminology change).

Clearly all aspects of the mapping must be complete and two-way. The mapping must support both:

— population of a shared datastore (an implementation of the IM) with data extracted from an application, and

— use of data from the shared data store in an application.

6 Process overview

Issue: at some point a decision will have to be made regarding the detailed documentation of the integration and mapping processes – should this be standardized, or (as in the case of STEP) documented and published as one or more SC4 Standing Documents?

6.1 IDEF0 diagrams

This section presents IDEF0 activity diagrams for the mapping and integation process.

Author’s note: these diagrams are incomplete. There are presented “as is” (taken from presentation N256) without additional documentation.

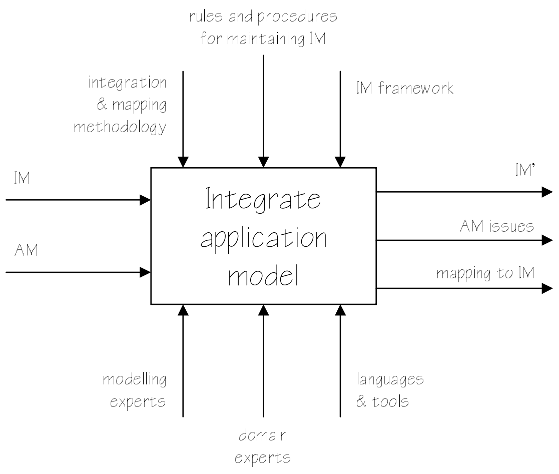
rules and procedures for maintaining IM
integration & mapping methodology
IM framework
IM
AM
Integrate application model
IM'
AM issues
mapping to IM
modelling experts
domain experts
languages & tools

Figure 6 – process overview (A-0)

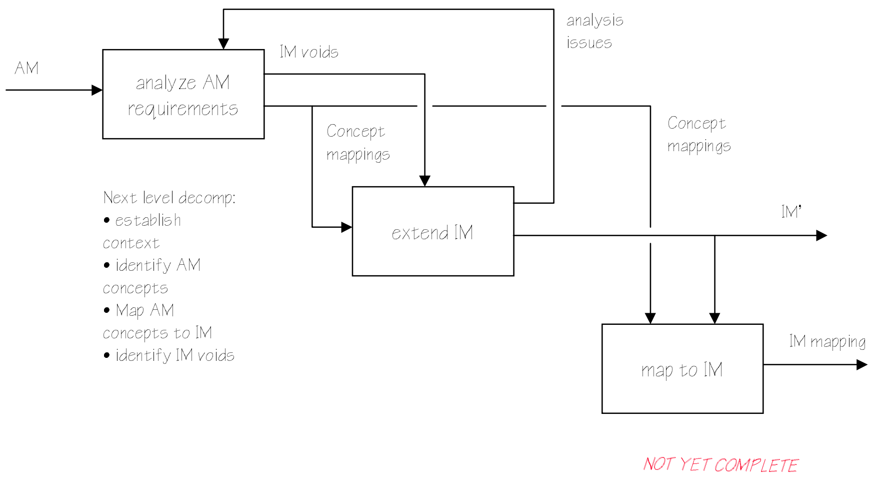

Figure 7 – process overview (A0)

6.2 Establishing the “context” (activity scope)

An important aspect of the mapping/integration process is to be able to associate data with a context (the activity/ies that create and/or use it. This is part of the association between the IM and a particular application data model.

NOTE “Context” is not really the right word to use here … although this is the same as *application context* in STEP. It is probably better to distinguish for a give application data model its *data scope* (the data elements that it contains) and its *activity scope* (the business processes or activities that it supports). The *activity scope* is what is meant here by “context”, and provides a link between the data model and an activity model.

There are a number of options for representation/characterization of the activity scope. The first is equivalent to that used in STEP APs – describe the scope in terms of an activity model (AAM), and identify the activity through population of data model elements equivalent to STEP’s application_context_schema. The second option is to make the link to the activity scope, by associating the data elements in a view with the class(es) of activity that they are involved in.

Issue for data architecture (not methodology): If activity scope is to be captured, does this go in the IM itself, or as part of standard “views” on the IM.

6.3 Documenting mappings

To be added in version 0.2 – there are bullet points in presentation N256, reproduced below.

Issue:

— Document IM ↔ AM mapping in one STEP

— Document separately:

- IM subset
- Model transformation(s)
- Terminology

STEP AP paradigm is a mixture of the two (IR - AIM - ARM)

Best choice: based on reusability

Recommendation:

— separate IM ↔ AM mapping into components and document each one

IM → IM subset

— is EXPRESS USE FROM sufficient?

IM subset → AM

— model transformation (several types)

— terminology change

N254 assumes separate mappings:

— projection/encoding (structure)

— IM subset → Application View

— terminology change

— Application View → Application Data Model

This needs to be reviewed/discussed - doesn’t seem to be consistent (is the AV in this case useful?)

Language for mappings:

— EXPRESS-X, MT, others?

7 Comparison with other SC4 (STEP) methods

To be added in version 0.2.

8 Conclusions and next steps

This document presents an initial, incomplete overview of the integration and mapping methodology that is a component of the SC4 data architecture. Following presentation and discussion at the WG10 workshop in London (April 1999) I will prepare a revised draft (time and resources permitting!) for discussion at the Lillehammer meeting.